The Intel® Random Number Generator

Intel Platform Security Division

CONTENTS

Introduction

In 1999, Intel will introduce a silicon-based random number generator (RNG) integrated into IA platforms for use by applications. Intel plans to incorporate the feature across its motherboards and chipsets, starting with desktop systems. The RNG is the first of Intel's family of primitives to be released for data and communications protection within the PC hardware.

The purpose of this application note is to introduce the RNG concept, to provide an overview of possible RNG applications, to discuss the quality of this new feature, and to highlight the value provided by the Intel® RNG.

Overview

What Is a Random Number Generator?

To define the concept of a RNG, it is first necessary to understand the idea of randomness. While the thought of randomness is typically associated with unpredictability, chance, and luck, mathematics provides a precise definition of randomness that can then be applied towards the evaluation of an RNG. Therefore, random numbers within the context of this paper refers to "a sequence of independent numbers with a specified distribution and a specified probability of falling in any given range of values" (*The Art of Computer Programming, Volume 2*, Donald E. Knuth). As a result, the ideal random number generator will provide a stream of uniformly distributed, non-deterministic, independent bits over an infinite data set. Because the mathematical evaluation of randomness is difficult, it is only possible to use statistical analyses on sample data sets to detect characteristics that point to "non-randomness."

RNGs can be implemented either in hardware or in software. Random number generation performed by software utilizes a mathematical algorithm that produces a sequence of independent numbers following a uniform distribution. However, this sequence is deterministic given the algorithm and the initial number, or seed. Because of these properties, software RNGs are also commonly referred to as pseudo-RNGs (PRNGs). While it is possible to implement a mathematical algorithm in hardware and call this part a "hardware random number generator," these RNGs also qualify as pseudo-RNGs because they require a seed and produce a deterministic sequence of numbers. True random number generation in hardware depends upon the random characteristics of physical elements; for example lava lamps, radioactive decay of atomic nuclei, or noise from a resistor or diode. There is no set sequence or seed with a hardware RNG, which also makes it useful as a seed for a pseudo-RNG. References to hardware RNGs in the remainder of this paper will refer to those that support true random number generation.

Applications for RNGs

A number of uses for random numbers has evolved over time. The following list provides some examples of random number applications:

Entertainment: Modern lotteries and gambling machines are all based on the use of random numbers. Video games use random numbers to influence intelligence heuristics or to vary game play.

Music and Graphics Composition: By interweaving content with random bits, the result can be used in image processing and reconstruction, electronic music composition, and 3-D texturing techniques for computer graphics.

Simulation and Testing: Complex scientific and financial models use random numbers for simulation. Many modern artificial intelligence applications require random data to determine classification accuracy or neural network behavior. Software manufacturers use random data to test programs and algorithms to detect bugs.

Equation-Solving: Mathematicians use random numbers to help them solve complex equations.

Cryptography, Digital Signatures, Protected Communication Protocols: Because of computer security, there is a growing interest in random number generation due to their value in key generation for cryptography, digital signatures, and protected communication protocols. At the base of these techniques used to secure data and data transmissions lies key generation, which requires the production of secret, unguessable keys. Hence, key generation

depends on an RNG to provide the required entropy to make the key indeterminable. Increased computer security enables businesses to take advantage of enhanced network services such as remote access and e-Business. With enhanced computer security, consumers obtain better and more convenient access to electronically available products and services such as shopping on-line, banking at-home, and downloading protected content.

Hardware vs. Software RNGs

Because PRNGs employ a mathematical algorithm for number generation, all PRNGs possess the following properties:

- A seed value is required to initialize the equation.
- The sequence will cycle after a particular period.

Therefore, application developers who require non-deterministic output from its PRNG must take pains to provide an unguessable seed value and an algorithm with a period that is sufficiently long. Methods to incorporate entropy into the seed include using user mouse movements, keystrokes, or key timings. However, system interrupt and event handling within different systems have been known to reduce the effective randomness of these sources. Furthermore, the application developer must verify that the PRNG output contains no correlation or bias. PRNG algorithm development requires considerable knowledge of PRNG theory. While strong algorithms have been developed, some require royalties for use. Consequently, the software manufacturer's cost for use of a strong PRNG includes the development cycle of the PRNG or the licensing of somebody else's pre-made algorithms. In either case, the software manufacturer must possess enough understanding of the PRNG implementation to prevent its properties from producing undesirable results for its users.

Hardware RNGs are available in the form of add-in PCI cards, serial or parallel port plug-in devices, and application-specific integrated circuit (ASIC) components for mounting onto a printed circuit board. They are non-deterministic by nature – no algorithm can be used to predetermine subsequent bits. Thus, hardware RNGs are not susceptible to intrusion by algorithm disassembly or disclosure. The property of non-determinism has been shown to be especially important in specific RNG applications such as government-sponsored lotteries, certain scientific and financial modeling techniques, and computer security technology such as cryptography and digital signatures. While the drivers may be specific to a certain operating system, a hardware RNG is potentially available to an application in any operating system, or even to BIOS.

With the use of a hardware RNG, software manufacturers do not need to hire developers with PRNG expertise nor pay fees for a third-party PRNG solution. However, software manufacturers still need to select a hardware RNG with a quality design and implementation from a reliable vendor. A quality design and implementation includes engineering to prevent failure, thorough testing to detect defects and non-random output, and software support for easy integration into applications. A reliable vendor ensures that the product remains available in the market and product adoption grows quickly.

The Integrated Platform Intel RNG

Design

Selecting a hardware RNG with a quality design is critical to the performance of the overall solution. Learning the quality of the design involves discovering the design considerations used, evaluating any experimental testing conducted, and measuring the manufacturing expertise of the vendor.

Intel's design was made following two key paradigms. First, the RNG should be based off a source that is truly random and based on properties native to silicon circuitry. Second, the design should preserve this randomness while providing the needed performance and circuit stability across temperature and other environmental effects. In order to meet these criteria and others, Intel's design uses white noise as the random source, then processes the data through a variety of circuit elements. These elements were chosen and configured so that the sampling frequency does not allow any relation between successive bits generated and provides a stream of bits with an even duty cycle. For an RNG, the duty cycle is the ratio of 1's to the total number of bits generated, and should be equal to 0.5.

Performance and robustness to environmental changes were measured in simulation and the appropriate modifications made.

Additional criteria used to evaluate the design include use of a figure of merit and monitoring the stream of bits provided. The figure of merit is a theoretical calculation that determines the sampling rate of the white noise source required to incorporate randomness. At each stage, extensive physical and statistical testing was performed to evaluate the randomness and suitability of the final design. To gain further confidence in the design and identify any remaining issues, the RNG was evaluated by independent third-party experts.

After making the theoretical design decisions, a quality RNG must stand up to experimental testing. In order to ensure the best possible solution, Intel tested the circuits not only through simulation but also through silicon testing of several circuit variations, especially at the extremities of specified temperatures and voltages. This level of testing ensures that simulation inaccuracies do not compromise the final solution and that it is chosen from a pool of strong candidates.

Strong manufacturing expertise is also critical to the quality of the RNG solution. Intel's vast experience in high-performance manufacturing was put to good use for the RNG. Design for manufacturing (DFM) considerations ensure that the RNG can be manufactured in a cost-conscious, high-throughput environment, and that it is robust against processing variations. Furthermore, the layout of circuits was carefully conducted to preserve randomness despite these physical variations.

The above considerations are just a few of those used to ensure the integrated, platform Intel RNG is a quality component for use by all applications, particularly in cryptography, digital signing, and protected communication protocols. Intel's design and manufacturing expertise sets it apart as a reliable supplier of this fundamental component in the platform.

Component Validation

In addition to testing of the silicon for circuit and interface defects, Intel performs a series of statistical tests to ensure the output adheres to the requirements of randomness by showing no bias or correlation. Sample data sets are collected from individual units and from lots for statistical testing prior to shipment. Also, random parts are selected for more extensive testing of bias and correlation as described below.

The presence of bias can be determined by calculating the average of the bits within a sample. Theoretically, an RNG with no bias should produce bits such that the average of the bits over an infinite data set equals 0.5 because each bit should have a 50% probability of being either a 0 or a 1. Because hardware RNGs rely on physical fluctuations, the average calculated on a sample will likely differ from 0.5 slightly. Instead, the averages of various sample sets should fall within specified bounds such that it is possible to be statistically confident that the ideal can be approached over the infinite set. These bounds are defined by 2 * the standard error which provides a 95% confidence interval around the mean. Table 1 presents test results from three 10-Mbyte samples collected from the Intel RNG under typical and extreme conditions. If the calculated average is less than plus or minus twice the standard error, then bias is considered to be undetectable within the sample. Equivalently, the test passes if the results are not statistically significant at 5%.

There are a few ways to measure the level of correlation between bits within a data set. One way to determine the presence of correlation is to calculate the correlation coefficient for specific pairs of bits, depending upon the order of correlation required. For example, first-order correlation calculated for a data stream represented by x_1, x_2, x_3, x_4, x_5, examines successive pairs of bits such as x_1x_2, x_2x_3, x_3x_4, etc. Second-order correlation examines pairs of bits such as x_1x_3, x_2x_4, x_3x_5, etc. While the explanation of the method of calculating the correlation coefficient is beyond the scope of this paper, a complete discussion can be found in *The Art of Computer Programming, Volume 2* by Donald E. Knuth.

The first-order correlation bears the most significance, and indicates possible patterns within the bits. Over the infinite data set, the correlation coefficient for the first order should equal 0 if the bits are independent. Again, this calculation on a sample collected from a hardware RNG will vary from 0 slightly, but the use of calculated bounds allows statistical confidence that the ideal can be approached over the infinite set. Table 2 provides test results from

three 10-Mbyte samples collected from Intel RNG for correlation detection under typical and extreme conditions. If the calculated correlation coefficient is less than plus or minus twice the standard error, then correlation is considered to be undetectable within the sample. Equivalently, the test passes if the results are not statistically significant at 5%.

Table 1. Bias Detection for 10-Mbyte Samples (room temperature)

Unit	Sample data collected at 25 °C and V_{CC} = 3.3 V	Sample data collected at 100 °C and V_{CC} = 2.9 V	Sample data collected at 0 °C and V_{CC} = 3.6 V
Calculated Average	0.500025	0.500073	0.499927
2 * Standard Error	0.000102	0.000110	0.000110
Test Passed?	Yes	Yes	Yes

Table 2. First-Order Correlation Detection for 10-Mbyte Samples (room temperature)

Unit	Sample data collected at 25 °C and V_{CC} = 3.3 V	Sample data collected at 100 °C and V_{CC} = 2.9 V	Sample data collected at 0 °C and V_{CC} = 3.6 V
First-Order Correlation	0.000145	–0.000007	–0.000047
2 * Standard Error	0.000204	0.000221	–0.000221
Test Passed?	Yes	Yes	Yes

Benefits to Application Developers

The Intel RNG strengthens any application or service which currently uses a software-only RNG. With an Intel Security Driver that supports existing security interfaces, application developers can easily take advantage of the hardware-based Intel RNG to enhance the security of their applications. No programming changes are required for applications that currently use pseudo-random number generation through existing security interfaces. However, applications that statically bind to a library providing the PRNG function will need to be re-linked. Unlike other proprietary hardware RNG solutions, no add-in boards or extra devices are required. All future IA platforms will include the RNG feature; therefore, software manufacturers can trust its availability across a widely-deployed infrastructure. Furthermore, Intel will provide developers with reference code to accelerate implementation time for new applications.

Tests for Randomness

Diehard Tests

Before computers, scientists typically rolled dice, tossed coins, or drew cards to provide random numbers. With the introduction of computers, the scientific community pushed the development of mathematical algorithms for random number generation to be used in scientific simulation, software testing, and mathematical modeling. As these algorithms were developed, problems arose from using algorithms that produced sequences containing patterns or that generated certain numbers more than others. To test the strength of various algorithms, Dr. George Marsaglia of Florida State University invented the Diehard test suite which contains a battery of statistical tests focused on weeding out PRNGs that have a short period, patterns within the bits, and non-uniform distribution of numbers.

While the Diehard tests target the possible shortcomings of PRNGs, the lack of an industry benchmark for characterizing performance of hardware RNGs has led to the popularity of Diehard for testing the output of hardware RNGs. Consequently, it is important to note that Diehard does not help determine reliability, throughput, tolerance to outside noise, or other physical characteristics important in reviewing hardware RNGs.

Intel has performed testing with Diehard using the output from the hardware RNG. Appendix C contains sample test results from one of the runs of the Diehard test suite performed on a 10-Mbyte sample. A full understanding of the Diehard output requires detailed knowledge of statistics. However, passing the Diehard tests is not very well defined since Dr. Marsaglia does not provide concrete criteria. An individual test can be considered passing if the p-value is between 0.025 and 0.975, forming a 95% confidence interval around the theoretical value specified within the test. However, to evaluate a data sample against the entire test suite requires consideration of all 250 p-values that are generated and the calculation of the probability that the entire suite passes with 95% confidence. This calculation yields a 95% confidence interval of 0.0001 and 0.9999 for the p-value. Therefore, the RNG fails the Diehard tests if there is a p-value greater than or equal to 0.9999 or less than or equal to 0.0001. It can be concluded that the output in Appendix C shows no abnormalities and all tests were completed successfully.

Federal Information Processing Standards

Federal Information Processing Standards (FIPS) are issued by the National Institute of Standards and Technology (NIST), the body responsible for developing technical standards and guidelines for federal information resources. These standards and guidelines are approved by the Secretary of Commerce for use throughout the federal government to protect sensitive, unclassified information. FIPS 140-1 covers security requirements for cryptographic modules and specifies recommended statistical tests for random number generators. Each of the tests is to be performed on 20,000 consecutive bits of output from the random number generator. If any of the four tests fail, then the random number generator is deemed to be in an error condition. The tests specified by FIPS 140-1 are as follows:

The Monobit Test:

1. Count the number of ones in the 20,000 bit stream. Denote this quantity by X.
2. The test is passed if $9{,}654 < X < 10{,}346$.

The Poker Test:

1. Divide the 20,000 bit stream into 5,000 contiguous 4-bit segments. Count and store the number of occurrences of each of the 16 possible 4-bit values. Let f(i) be the number of each 4-bit value where i is between 0 and 15.
2. Evaluate the following equation:

$$X = \frac{16}{5000} * \sum f(i)^2 - 5000$$

3. The test is passed if $1.03 < X < 57.4$.

The Runs Test:

1. A run is defined as a maximal sequence of consecutive bits of either all ones or all zeros. Using a 20,000 bit block, the incidences of runs (for both consecutive zeros and consecutive ones) of all lengths (>=1) in the sample stream is counted and stored.
2. The test is passed if the number of runs that occur (of lengths 1 through 6) is each within the corresponding interval specified below. This must hold for both the zeros and ones; that is, all 12 counts must lie in the specified interval. For the purpose of this test, runs of greater than 6 are considered to be of length 6.

Length of Run	Required Interval
1	2,267 – 2,733
2	1,079 – 1,421
3	502 – 748
4	223 – 402
5	90 – 223
6+	90 – 223

The Long Run Test:

1. A long run is defined to be a run of length 34 or more (of either zeros or ones).
2. On a sample of 20,000 bits, the test is passed if there are **no** long runs.

Using data collected from the Intel platform RNG, the following table provides results of five consecutive runs of the FIPS 140-1specified tests. Output from the Intel RNG passes all four statistical tests successfully.

Table 3. Results of FIPS 140-1 Specified Tests (room temperature)

Run #	Monobit Test	Passed Test?	Poker Test	Passed Test?	Runs Test		Passed Test?	Long Run Test	Passed Test?
1	X=9,940	Yes	X=10.69	Yes	Zeros 1: 2,421 2: 1,223 3: 681 4: 335 5: 162 6+: 147	Ones 1: 2,493 2: 1,251 3: 603 4: 311 5: 160 6+: 151	Yes	No long run	Yes
2	X=9,879	Yes	X=26.16	Yes	Zeros 1: 2,415 2: 1,244 3: 643 4: 326 5: 151 6+: 165	Ones 1: 2,435 2: 1,228 3: 638 4: 340 5: 152 6+: 150	Yes	No long run	Yes
3	X=9,898	Yes	X=9.52	Yes	Zeros 1: 2,560 2: 1,320 3: 594 4: 262 5: 155 6+: 139	Ones 1: 2,513 2: 1,232 3: 612 4: 347 5: 151 6+: 176	Yes	No long run	Yes
4	X=9,895	Yes	X=11.78	Yes	Zeros 1: 2,553 2: 1,239 3: 583 4: 334 5: 162 6+: 149	Ones 1: 2,482 2: 1,262 3: 679 4: 287 5: 158 6+: 151	Yes	No long run	Yes
5	X=10,030	Yes	X=10.58	Yes	Zeros 1: 2,448 2: 1,246 3: 621 4: 338 5: 163 6+: 157	Ones 1: 2,502 2: 1,238 3: 603 4: 310 5: 163 6+: 157	Yes	No long run	Yes

Conclusion

The Intel Random Number Generator has been designed with strong attention to quality and manufacturability. Tests on the RNG show excellent performance to various cryptographic evaluations. Two examples of strong performance and suitability for cryptographic applications are the passing results from Diehard and FIPS 140-1 tests.

References

[1] FIPS 140-1, *Security Requirements for Cryptographic Modules*, Federal Information Processing Standards Publication 140-1, U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, Virginia, 1994.

[2] Robert Davies, *True Random Numbers*, http://webnz.com/robert/true_rng.html.

[3] D. Eastlake, S. Crocker, and J. Schiller, *Randomness Recommendations for Security*, RFC 1750, 1994.

[4] Carl Ellison, Cryptographic Random Numbers, Draft P1363 Appendix E, http://www.clark.net/pub/cme/P1363/ranno.html.

[5] Donald E. Knuth.: *The Art of Computer Programming, volume 2: Seminumerical Algorithms*. Addison-Wesley, Reading, MA, 2nd edition, 1981.

APPENDIX A

TECHNICAL SPECIFICATION

Fractional probability of excess 1s†:	$\pm 110.5 * 10^{-6}$ (max)
First Order Correlation Coefficient††:	$\pm .0221 * 10^{-6}$ (max)

NOTES:

† This calculation provides the probability of bias present within a 10-Mbyte sample.

†† This calculation provides the probability of first-order correlation present within a 10-Mbyte sample